Extensions – Introduction

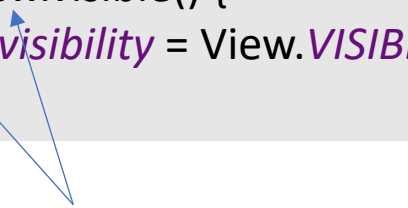
- Similiar to Swift, C# and Gosu extensions in Kotlin expand the functionality of standard classes.
- **Kotlin** supports **extensions functions** as well as **extension properties**.

Extensions – Introduction

How to we declare extension functions ?

Just write a function as you would normally, and put the name of the class (receiver) before the function name seperated by a **“.“**

```
fun View.visible() {
    this.visibility = View.VISIBLE
}
```



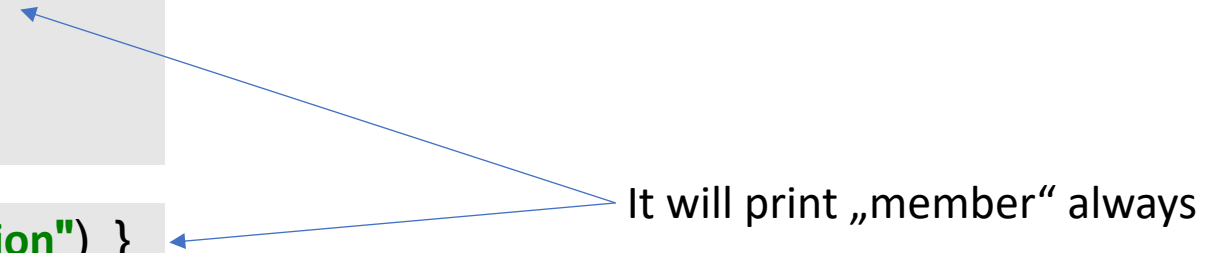
Receiver class (View)

Extensions – Rules – Members always win

Member funtions always win.

```
class C {
    fun foo() {
        println("member")
    }
}
```

```
fun C.foo() { println("extension") }
```



Extensions – Rules – Nullable Extensions

It is possible to define extension functions for a nullable types

```
fun Any?.toString(): String {
    if (this == null) return "null"
    return toString()
}
```

Extensions – Rules – Static Extensions

You can define extension function on class level instead of instance level.

```
class Foo{

    companion object

    fun sayHello() = "Hello"

}
```

```
fun Foo.Companion.sayBye() = "Bye"
```

Extensions – Properties

As mentioned in the beginning, you can also define extension properties for classes. The only restriction is that you can‘t initialize the property directly. You have do explicitly define the getter and setter for it.

```
val Foo.bar = 1
```

// error: initializers are not allowed for extension properties

```
val Foo.bar: Int
    get() = 1
```

Extensions – Dispatching

```
interface Loggable {

    val Any.LOGGER: Logger
        get() = Logger.getLogger(javaClass.name)

    fun Any.logI(message: String){
        LOGGER.log(Level.INFO,message)
    }

    fun Any.logE(message: String, error: Throwable){
        LOGGER.log(Level.SEVERE,message,error)
    }
    //...
}
```

Dispatch Receiver

Extension Receiver

Extensions – Reified and Inline

```
inline fun <reified T : Activity> Activity.navigateTo(intentParameters: Map<String, Serializable>) {
    val intent = Intent(this, T::class.java)
    intentParameters.forEach { s: String, serializable: Serializable ->  intent.putExtra(s, serializable)}
    startActivity(intent)
}
```

```
navigateTo<SessionActivity>(mutableMapOf())
```